Evaluation The Game Of DeviousMUD, Its Design, Implementation And Usage

by Finchley Alexander Harwood

FILE	23673_FINCHLEY_ALEXANDER_HARWOOD_EVALUATION_THE_GAME_OF_DEVIOUSMUD__ITS_DESIGN__IMPLEMENTATION_AND_USAGE_312866_1604402070.PDF (5.12M)		
TIME SUBMITTED	11-DEC-2020 04:19PM (UTC+0000)	WORD COUNT	3677
SUBMISSION ID	139285646	CHARACTER COUNT	17742

Evaluation: The Game Of DeviousMUD, Its Design, Implementation and Usage

Finchley A. Harwood

School of Applied Computing

University of Wales Trinity St. David

December 2020

Abstract

DeviousMUD, is a text-based Role Playing Game (RPG) set in a collection of rooms which make up a 'dungeon'. The implementation of an Agile development process is evaluated. This followed by an evaluation of the design, implementation and testing of the database and java code. Finally, there is a discussion of some of the author's own contributions to the project and their personal reflection on how the project went.

Contents

1 Introduction **3**

2 The Agile Process **3**
2.1 User Story Mapping 3
2.2 Sprint Planning 4
2.3 Daily Stand-up 5
2.4 Sprint Retrospective and Review 5
2.5 A Self-Organising Team 6

3 Database Design, Implementation and Testing **7**
3.1 Database Design 7
3.2 The Database 8
3.3 Connections and Testing 8

4 Software Design, Implementation and Testing **8**
4.1 Software Design 8
4.2 Software Implementation 9
4.3 Testing 9

5 The Author's Contribution To The Project **10**
5.1 The 'Move' Story 11
5.2 The 'Attack' Story 11
5.3 The 'Boss Room' Story 12

6 Reflection And Summary **13**
6.1 Personal Reflection 13
6.2 Reflection On The Importance Of Agile 15
6.3 Summary 15

1 Introduction

DeviousMUD, is a text-based Role Playing Game (RPG) set in a collection of rooms which make up a 'dungeon'. The aim of the game is to find the 'boss room' and defeat all non-playable characters (NPCs) contained within it. On their journey throughout the dungeon, the player can interact with 'chests' and NPCs, obtaining information about the dungeon itself, and gaining items and gold. The player's gold amount is recorded to a score board at the end of their play through.

The following report will evaluate the overall implementation of the game, with a specific focus on the author's own contribution. The game was constructed over four days, with the first day spent designing and the final three days spent coding and testing.

2 The Agile Process

The work was completed within an Agile setting, this meant that the team treated the four day period of work like a 'sprint' in the SCRUM framework [1]. The following section will discuss the team's implementation of the SCRUM framework.

2.1 User Story Mapping

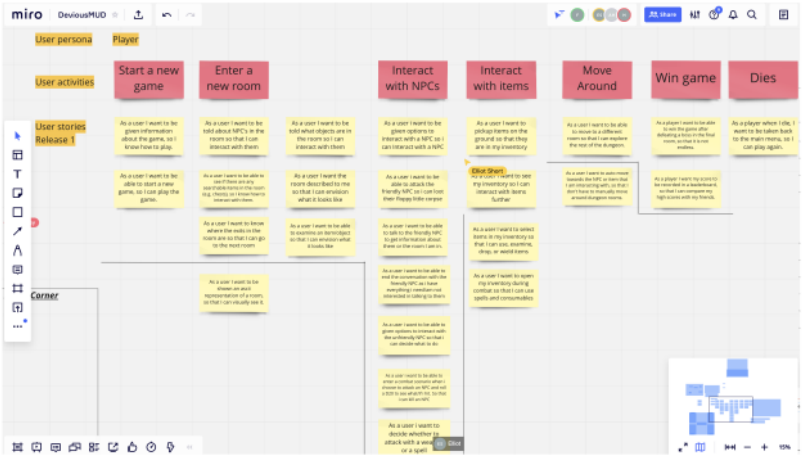

Figure 1: A screenshot taken from the User Story Mapping session carried out by the team. This was completed using the web application 'Miro', which acts like a virtual whiteboard.

On the first day of the sprint the team organised a conference call in-which they discussed the project and how they might go about completing it. Thereafter ensued a collaborative User Story Mapping [1] session in which the team decided what the key user activities were and came up with User Stories to reflect these activities. A screenshot of this session can be seen in Figure 1.

User story mapping was a key part in the design process. It meant that the team were clear on which User Stories they were going to complete and everyone had a shared vision of what the end product would look like.

2.2 Sprint Planning

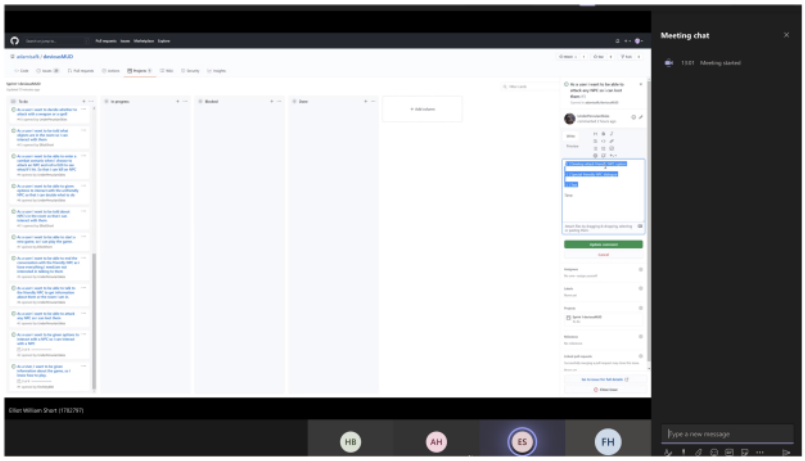

Figure 2: A screenshot taken from inside Microsoft Teams, showing the Sprint Backlog on the Github repository. In this specific example the team are generating sub-tasks for a specific User Story.

After the User Story Mapping session, the team went into Sprint Planning. Each User Story was taken through the same process: firstly, the User Story was placed on the GitHub projects 'Sprint 1' Kanban board; secondly, the User Story was broken down into sub-tasks (see Figure 2); and thirdly, an estimate was made for the User Story using Planning Poker and the Fibonacci Sequence (see Figure 3).

[1] This is the process of turning requirements into User Stories and mapping out a user's journey through a program [2].

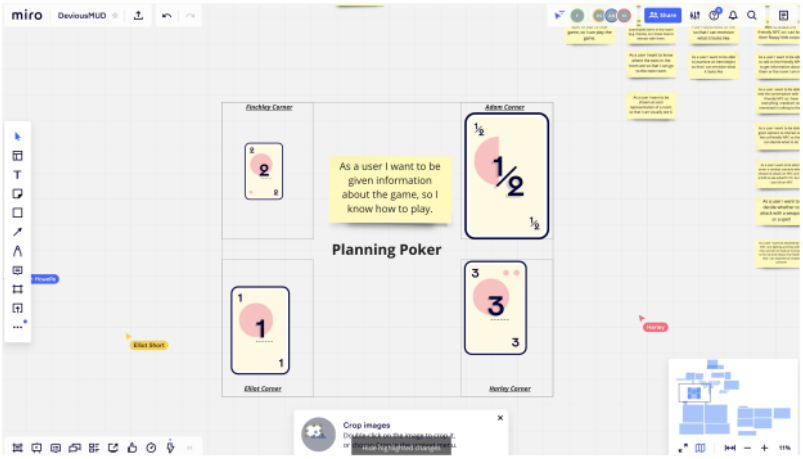

Figure 3: A screenshot showing the team's planning poker session, carried out within Miro. After placing down their estimates, the team came to a collective decision on the size of each of the User Stories.

The Sprint Planning meeting was extremely useful. The team found themselves refining the User Stories which were generated in the User Story Mapping session, and removing User Stories that were already covered by other User Stories. The team also decided on a sprint goal of *'Create a simple text-based RPG, connect this to a database for storing maps and recording player scores'*.

2.3 Daily Stand-up

Every morning at 09:00 hrs the team had their 'Daily Stand-up'. During this meeting they were able to sync up their work and plan for the day ahead. The Daily Stand-up was facilitated by Elliot Short, the team's elected Scrum Master. Elliot was responsible for organising the meetings and making sure that everyone was heard.

2.4 Sprint Retrospective and Review

At the end of the sprint the team held their 'Sprint Retrospective'. This meeting provided an open and democratic space for the team to discuss how they felt their work had gone. The Screenshot of Figure 4 was taken during this meeting, and shows three columns. Team members placed notes in the columns describing what they thought went well, want did not go well and what could be improved.

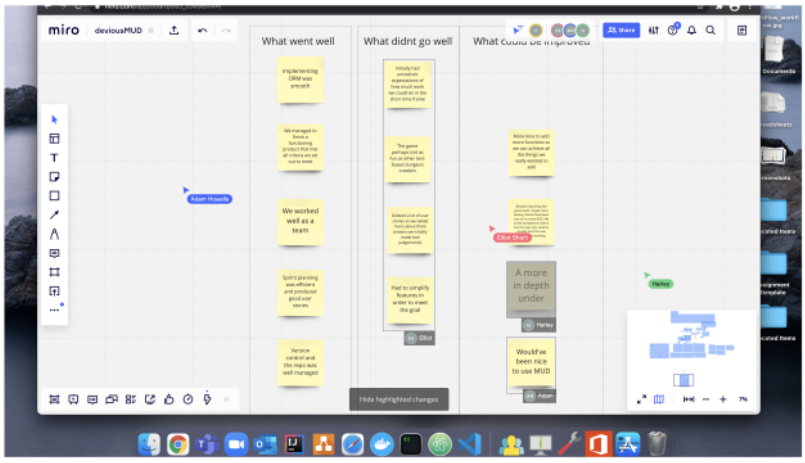

Figure 4: A screenshot showing the team's Sprint Retrospective, carried out within Miro. Everyone placed down something they thought had gone well, something that didn't go well and something they thought could be improved.

It is a shame that the team were not able to showcase their work in a 'Sprint Review' meeting. However, one of the team members, Harley, created a Youtube video [2] to demonstrate a play-through of the game. When the video was shared with other members of the student cohort, the team was able to *simulate* a Sprint review, in that they received feedback on the work they had completed, which is essential to working in an Agile way.

Both the Sprint Retrospective and (sort of) Sprint Review were useful to the team. If the team were to complete another sprint, they would have an invaluable set of information that they could use as a guide for improvement.

2.5 A Self-Organising Team

Ac self-organising team, is an essential aspect to the implementation of Agile [3]. The team who developed 'deviousMUD' were self-organising in the fact that every person assumed a role that reflected their ability and experience. In the author's own view, the team were a microcosm of a larger scale Agile team. One of the team members was experienced with using databases and software development, and they assumed the role of a DBA/ Senior Devloper. The two lesser experienced members of the team assumed the roles of Junior Developers. The author assumed the role of a 'tech-lead', drawing up UML diagrams, having a good understanding of the project as a whole, and using their experience to

[2] Please contact either the author, or Harley for a link to this video.

help the 'Junior Developers' when needed. It must be noted that this is not to say that each member of the team worked on only a specific part of the project, but it is to say that there was a general focus in certain areas from certain members of the team.

3 Database Design, Implementation and Testing

3.1 Database Design

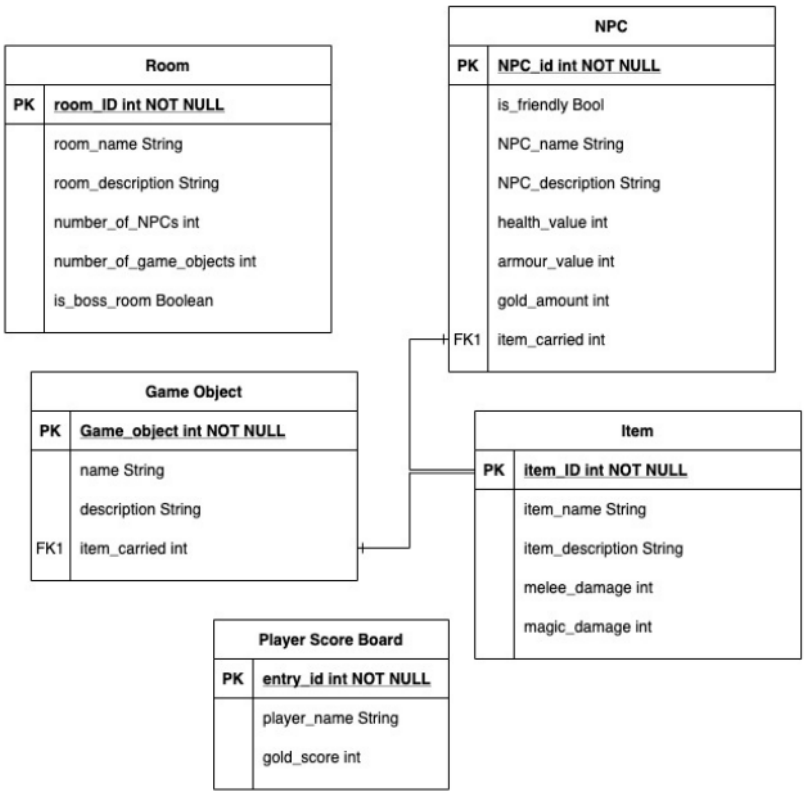

Figure 5: The database schema for deviousMUD. There are five tables: room, game object (chest), item, NPC and score. Every NPC and chest carries one item, which the player can 'take' from them. Diagram created by the author, and can also be found on the GitHub repository under the wiki section.

On the first day of the sprint, after the sprint planning session, the team had a meeting to discuss how they would implement the database in deviousMUD. This discussion lead to the author creating the database schema of Figure 5. The lack of relationships between the Room table, and the NPC and Game Object (Chest) table is due to a design decision by the team. The team decided that they wanted to randomise the number of and the type of NPCs and chests in a room, to improve 'replayability'. This database schema is slightly out of date, because the two columns on the Room table referring to the number of NPCs and Game Objects are now redundant.

3.2 The Database

The team came to the collective decision of using 'MySQL' as their database management system. This is partly due to the fact that two of the developers already had MySQL installed on their machines. The majority of the team used the IDE 'DataGrip' to construct database tables and enter data into those tables. However, the author was more comfortable using MySQL's terminal shell to interact with databases. DataGrip produced an 'sqldump' file which was then stored in the repository. This could then be run in the macOS terminal [3] to update each of the developer's local databases when a change had been made.

3.3 Connections and Testing

The team decided, on the recommendation of the 'DBA', to use an Object Relational Mapping Tool (ORM) called 'Hibernate' to connect the database to objects in the code. Instead of directly interfacing with Hibernate, the team used the Java Persistence API (JPA) [4]. This meant that if the team decided to change to a different ORM, the transition would be relatively smooth.

The `JPAUtil` class in the `entity` package contains static helper functions which use JPA's `EntityManager` class to get data from the database. All of the SQL queries that exist in the code, exist in this file (this excludes queries that were used to check the data was implemented properly in the MySQL shell). The author and Adam Howells collaborated on this utility class during the second day of the sprint; the author wrote the majority of the corresponding unit tests (which can be found in the `JPAUtilTest` class), and wrote some of the functions in `JPAUtil`.

4 Software Design, Implementation and Testing

4.1 Software Design

On the first day of the sprint, the author constructed a UML diagram to represent each of the classes within the proposed game of deviousMUD. This

[3] The command for this can be found on the Github repository.

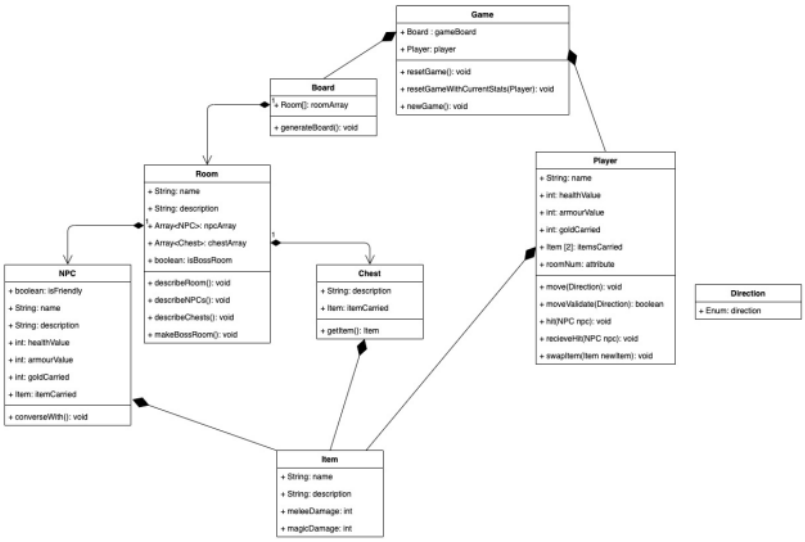

Figure 6: The UML diagram constructed by the author on the first day of the sprint. This diagram was used throughout the project as a guide for the developers.

diagram was used by everyone on the team as a guide to how the end product should work. Multiple team members commented on how useful this diagram was, and it is the author's opinion that the UML diagram contributed in a large part to the team being able to release a working product at the end of the sprint.

4.2 Software Implementation

The implementation of the code closely resembles the UML design constructed on the first day. Even when there are differences, the important thing to note is that *logically*, the UML design represents the overarching structure of the code. The code is a little bit messy, and the author would like to, if given more time, make a helper class which contained useful functions. They would also like to separate out some of the functions in the `Game` class to make them more testable. The diagram of Figure 7 is an attempt at representing the final structure of the code; attributes and functions belonging to classes are likely incorrect.

4.3 Testing

Testing is perhaps one of the places where the author is not completely satisfied with the result of the sprint. Significant efforts were made to provide unit test coverage for the code however, whether this is sufficient can be left to the

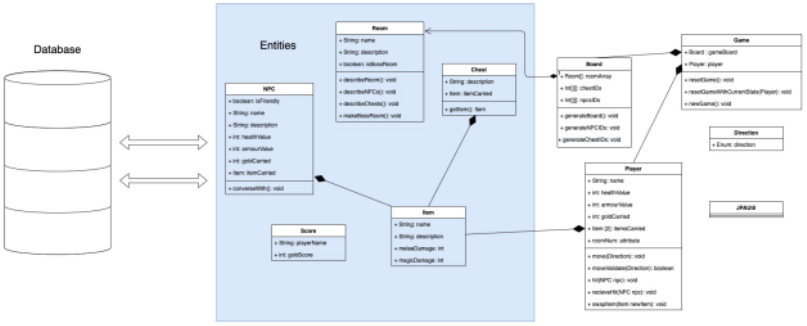

Figure 7: A UML-esque diagram showing the final structure of the code. Note: this is the author's own interpretation and it may not be the 'correct' way to go about constructing diagrams of this nature.

reader to decide. The author would like to point out that the ORM entities were not given unit tests because it was assumed that the database was functioning properly. Classes such as `Game`, `Board`, and `Player` have a number of unit tests, but perhaps not full unit test coverage in some cases [4].

5 The Author's Contribution To The Project

Each day of the sprint, the team entered a group call and either mob-programmed, or pair-programmed their way through each of the User Stories. It was incredibly useful to be able to instantly ask a member of the team for help on a specific task.

Whilst the author completed many user stories, and some of these will be discussed later on in this section, the author must stress that they were constantly on hand helping the less experienced members of the group. This means that if the reader views the commit log, they will find commits from the author in almost every part of the program.

It is difficult to estimate the amount of work that the author completed, but the screenshot of Figure 8 gives a rough idea of how much work they did. On one particular night the author worked a thirteen hour day to ensure that the project was completed on time. It is important to stress that the author did this because of their own passion for the project, and so it must be viewed as the author going above and beyond, rather than other members of the team under-performing.

[4] The author urges the reader to view the commit log for the testing folder on the github repository.

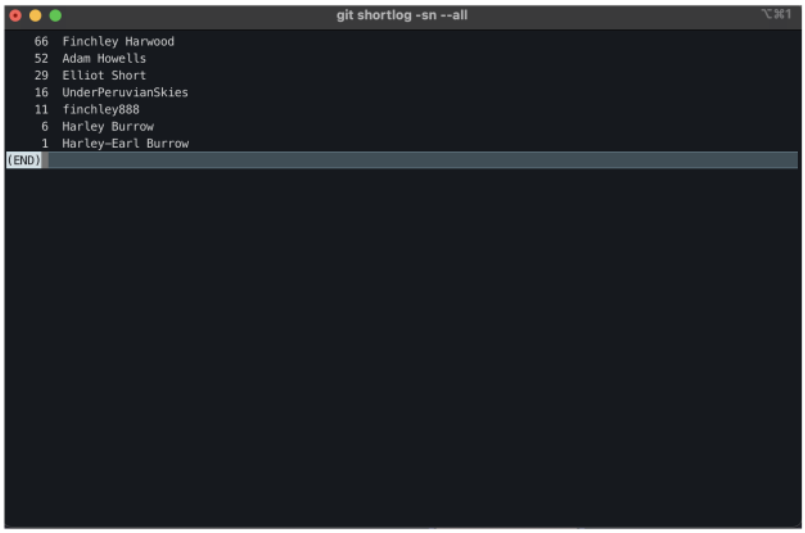

Figure 8: A screenshot taken of the result of running the command `git shortlog -sn --all` to view the number of commits by each team member. The author's contributions are under 'Finchley Harwood' and 'finchley888'.

5.1 The 'Move' Story

The 'move' story or *"As a user I want to be able to move to a different room so that I can explore the rest of the dungeon"* was completed by the author. The content of this story can be seen in the screenshot of Figure 9. The functions `moveValidate()` and `move()` under the class `Player`, `evaluateInput()` under the class `Game`, and the `Direction` enum class were all written by the author for the purposes of this user story. Unit tests for this user story can be found in the test class `PlayerTest`.

The acceptance criteria do not quite match what the user input actually is. To move, the player must type 'move ¡direction¿' where direction is either 'east', 'west', 'south' or 'north'. However, this is very a minor issue.

5.2 The 'Attack' Story

The 'attack' story or *"As a user i want to be able to enter a combat scenario when i choose to attack an NPC and roll a D20 to see what/if i hit. So that i can kill an NPC "* was a collaboration between the author and Harley. Harley wrote the functions `hit()` and `recieveHit()` in the `Player` class. These functions return a randomised amount of damage between 0 and the maximum magical or melee damage of the item that the player is currently wielding.

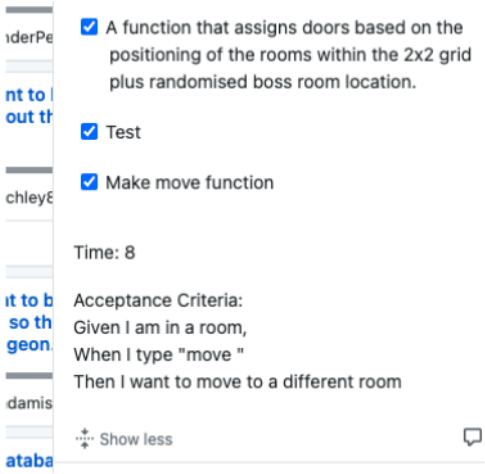

Figure 9: A screenshot of the contents of the 'move' story, which was completed by the author. This is taken from the Kanban board on the Github project repository.

The author combined Harley's functions in a function called `playerBattle()` in the `Game` class. This is called when the user enters the command "attack" in game. The user is then presented with a list of alive and dead NPCs in the room. They are only able to attack an NPC that is alive (this is governed by a indexed-by-room array of NPC health which is exists in the `Board` class). The author wrote unit tests which test that the NPC health array is functioning correctly; these can be found in the class `BoardTest` [5].

5.3 The 'Boss Room' Story

The 'Boss Room' story or *"As a player I want to be able to win the game after defeating a boss in the final room, so that it is not endless."* was a collaborative effort between the author and Adam. Adam handled the functionality of saving the player's score to the database at the end of the game, and the author handled the win conditions. The screenshot of Figure 11 shows the initial plan for the story. The author completed this story in the closing hours of the sprint, and so took the liberty of changing the win conditions slightly (with the permission of the rest of the team of course).

The author designed it so that the player must find there way to the boss room at the end of the dungeon and defeat all of the NPCs in there. The player can

[5]This class contains other tests written by the author, testing that the NPC health is reduced properly in a fight, and a test for a function used in the boss room.

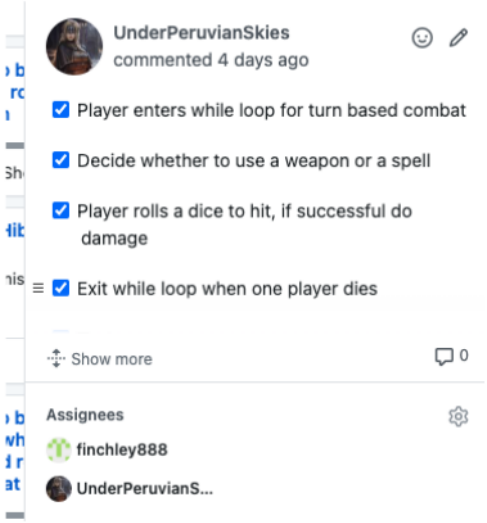

Figure 10: A screenshot of the contents of the 'attack' story, which was completed by the author and Harley. This is taken from the Kanban board on the Github project repository.

then type 'leave' and they will win the game. If the player types 'leave' in any other room or if some of the NPCs are alive in the boss room, then they cannot leave. A function was written for this purpose called `allNpcsInRoomAreDead()`, which is contained in the class `Board`. There are corresponding unit tests for this function in the class `BoardTest`.

6 Reflection And Summary

6.1 Personal Reflection

The author is immensely proud of the work that they and the rest of the team managed to complete in the four day sprint. Every member of the team put in as much effort as they could to produce a product that works. DeviousMUD is a playable game, whether it is enjoyable to play is another matter, but it is the author's opinion that it can be classed as a so-called Minimum Lovable Product (MLP).

There is always room for improvement, and this project is no exception. There are many things that could have gone better, any many features that would have been nice additions to the game. Three areas of improvement will now be discussed in relation to the three areas of assessment for this project. This is

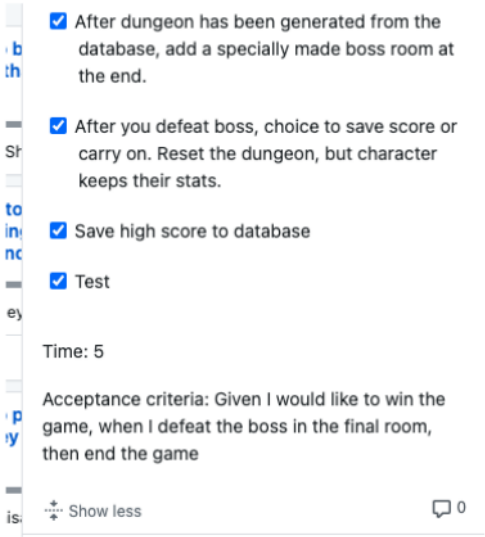

Figure 11: A screenshot of the contents of the 'boss room' story, which was completed by the author and Adam. This is taken from the Kanban board on the Github project repository.

not to say that there are not other areas of improvement, but for the purposes of being concise, only three will be discussed.

The first area of improvement is in relation to the agile process. The team set out to implement Behavioural Driven Development (BDD), by including 'Gherkin' 'Acceptance Criteria' in the User Stories. The implementation of this never really came to fruition, acceptance criteria were used as a guide in play testing, but it would have been nice to be able to use the acceptance criteria to write some actual 'Gherkin' tests.

The second area of improvement relates to the database implementation. It would have been nice for the team to use more of the skills that they have learnt when it comes to database design and implementation. The database is not very complex, and does not demonstrate the fact that the team could produce a far more comprehensive database. Although, for the purposes of this project it is perfectly adequate, and perhaps adding in more tables and relationships might be considered 'overkill'.

The third area of improvement relates to the implementation of software. The author believes that some of the code is a bit 'messy' and not particularly clean.

Where the author has written code they have tried to include java documentation and write unit tests, but in some cases this has been neglected by the rest of the team. This is perhaps due to time constraints, and it is hoped by the author that the team have done enough to demonstrate that they know how to write clean and testable code.

6.2 Reflection On The Importance Of Agile

The fact that the team were agile in their software development went a long way to ensuring the success of the project. Meetings such as daily stand-up, user story mapping, sprint planning and even the sprint retrospective helped the team to work efficiently and effectively together. Whether or not this project could have been completed if the team had tried a non-Agile, perhaps Waterfall-based, approach is unanswerable, but it appear that perhaps it would have been harder, with more time spent planning and less time spent coding.

6.3 Summary

To summarise, the game of deviousMUD is a text-based RPG dungeon-crawler. The team tasked with developing this game managed to complete it within a four day 'sprint'. In this sprint the game was designed, implemented and tested, through unit testing and manual play testing. The author feels that they could not have worked any harder than they did, and they feel that the same can be said for all of the other members of the team. The author would like to thank the reader, and hopes they enjoy playing the game.

References

[1] M. Hron and N. Obwegeser, "Scrum in practice: an overview of scrum adaptations," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[2] S. Foehrenbach and C. Heldstab, "User story mapping: The hands-on course," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2017, pp. 1217–1219.

[3] S. Licorish and S. MacDonell, "Self-organising roles in agile globally distributed teams," in *24th Australasian Conference on Information Systems (ACIS)*. RMIT University, 2013, pp. 1–11.

[4] L. DeMichiel and M. Keith, "Java persistence api," *JSR*, vol. 220, 2006.

Evaluation The Game Of DeviousMUD, Its Design, Implementation And Usage

GRADEMARK REPORT

FINAL GRADE

90/100

GENERAL COMMENTS

Instructor

Software design evaluation (30): 28

Database evaluation and implementation (30): 25

Coding and Testing (30): 27

Reflection (10): 10

An excellent report overall. It might have been nice to see a little more evaluation of the user story mapping, but a coherent write-up overall.

You have reflected upon the overall database design, however a little more evidence your design considerations would have been helpful.

Good evidence of teamwork, clearly you worked together well and formed good practices.

A very impressive implementation overall - clearly a significant amount of effort went into this and it was evidently a labour of love.

Ensure when creating user stories that the user is given a persona, e.g. call them "player" instead of "user".

The team has clearly put significant effort into learning about ORM and this has produced a nice real-world type solution. While we were expecting traditional

database queries, this is an appropriate (and better) alternative.

Some duplication of code which could have been separated into functions, e.g. the try/catch blocks in game.java.

Some of the game logic which is currently hard coded could have benefited from being in the database, this might have allowed extensibility, e.g. allowing different levels.

You have evidently made some effort to ensure the code is testable and, therefore, the tests are detailed and appropriate.

Very well done on an excellent implementation!

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16